E-mail Content Scanning with Exim 4

Complete Notes from the 2005 Exim Course

Updated Date: 14 July 2005

©2005 Tim Jackson (tim@timj.co.uk)

Introduction

Earlier parts of the Exim course reviewed how ACLs and conditions can be used in many powerful ways that don't involve checking the actual *content* of a message; for example, DNS-based blacklists (DNSBLs) and checks on the sender address. These checks are (usually) relatively 'cheap' in terms of resources, and can provide a powerful framework for protecting against spam, viruses and other unwanted messages. However, there is frequently a requirement for doing checks which extend beyond variables related to the message envelope and connection, and into the message body itself.

Here, we'll examine some of the issues involved in message content scanning, including:

- Content scanning at the MTA – pros and cons of this option
- Methods of implementing content scanning – some of the options available
- Software required to implement content scanning with Exim
- A brief overview of common external scanning software
- A look at the Exim Content Scanning Extension
- Some other considerations and example configurations

Basic Rationale & Considerations

The basic rationale to content scanning is usually obvious: to reduce the amount of spam and viruses transmitted by e-mail. However, performing such checks effectively does very often mean venturing into the message body.

Typically, content checks are more 'expensive' than checking the basic variables already mentioned, not least because they frequently involve external tools such as virus scanners. So, content scanning should be used as a last resort, rather than as the default way of blocking unwanted mail. It's likely to be much more efficient both from a bandwidth and CPU load point of view to, for example, check hosts delivering incoming messages against trusted DNS blacklists and reject as much as possible at SMTP RCPT stage, rather than allowing all messages to be received in their entirety and then make rejection decisions based on content.

Whilst most 'security' or other checks on inbound mail involve a certain amount of consideration of policy issues beyond merely technical considerations, content scanning has, perhaps, more than most. Some people are sensitive about their mails being examined and reported on, even if it is merely by a computer program. The many colourful variations of language and context can take their toll on any attempt to make sense of them by a computer. Furthermore, some care needs to be taken if content checks are not in fact to worsen the problems they are intended to solve.

Inbuilt methods for content scanning

There are two primary methods of implementing content scanning: at the SMTP protocol stage and after a message has been accepted. Using Exim's core built-in methods, you can implement scanning at the SMTP protocol stage using ACLs; specifically the DATA ACL where you can check, for example, the $message_body variable in a condition and make decisions based on it. Once a message has been accepted, Exim or Sieve filters can be used to affect the delivery of a message including making decisions based on its content.

Some common external content scanning software

Exim's standard internal ACLs and filters have already been discussed on this course, so I will focus primarily on the use of “external” content scanning tools, particularly the Content Scanning Extension for Exim which allows the use of many common pieces of content scanning software. Inevitably in these days of inboxes choked with junk, the primary concern of many people is stopping spam and viruses, so software typically used includes:

- SpamAssassin – a Free software spam checker
- Clam Antivirus – a Free software virus scanner
- Sophos – a commercial virus scanner, normally used together with 'sophie', a Free software daemon

Content scanning at the MTA

So, given that we are considering Exim, we need to look first at the implications of using an MTA such as Exim to implement content scanning. Let's not forget that the use of an MTA is not *required* by any means; in fact, it's unusual in some respects, since a mail *transport* agent is normally responsible solely for, as its name suggests, moving mail from one place to another without taking any action based on its contents. Indeed, in common with other elements of e-mail infrastructure and terminology, the analogy with traditional postal services holds true: their job is to transport mail from one place to another, not to open and inspect it along the way.

However, it's worth bearing in mind when considering the options available that many end-user software packages (MUAs) have content scanning options of one kind or another built in, whether it's simple filtering or Bayesian anti-spam learning and filtering.

Looking then at the advantages and disadvantages of content scanning at the MTA:

Advantages

- Centralised – easy to apply consistently across large user populations. In any large environment, deploying suitable content scanning software to end users can be a challenge, but with the constant changing and improving of techniques, maintaining up-to-date software for a large number of users may be at best impractical.
- Transparent – no need for end users to install or configure software, or to learn about content scanning. If there is such thing as an “average non-technical user”, he/she is probably not particularly interested in learning about spam filters for example – they just want a mailbox free of junk.
- Easy to maintain – all in one place. Modifications and upgrades can be instantly applied.

Disadvantages

- Confuses the role of an MTA, which normally only *transports* mail rather than taking action based on its content
- Can be a blunt instrument, takes control away from users (although this cuts both ways as mentioned earlier)
- Centralises burden on mail servers – resource usage needs to be considered. Content scanning (depending on what is required and what software is used) can potentially add a significant load overhead as opposed to merely delivering mail.

Two primary ways to implement content scanning

As touched on earlier, there are two primary methodology choices when implementing content scanning at the MTA:

- Accept-and-scan – accept the message at SMTP time, then process content and make decisions at a later point in time
- SMTP-time scanning – scan the message "live" during the SMTP session and do not even accept.

Each method has distinct characteristics along with advantages and disadvantages, which we'll look at in more detail. Before we carry on, one important thing to bear in mind at all time is that *a message can have multiple recipients*. It may sound obvious, but a lot of fruitless discussion often takes place making simplistic assumptions such as that there is a one-to-one relationship between messages and recipients (at least on any given server). This may often be the case in practice, especially with the widespread adoption of VERP-based mailing lists, but we cannot assume this (except in some unusual cases which will be discussed later).

Accept-and-scan

Because we accept the message at SMTP time, we have more flexibility over what to do with it. Using Exim's internal methods, we can use filters to make decisions on delivery. Decisions could range from bouncing the message, to discarding it, forwarding it or perhaps delivering to different folders according to the content (e.g. saving suspect spam to a separate folder to "normal" mail). Alternatively, if the use of software such as SpamAssassin is required, a router can be used which passes the message out to SpamAssassin and, depending on the result, typically either fails it (generating a bounce) or re-injects it into Exim, using a marker (such as the 'protocol' option) to prevent a scanning loop.

The primary advantages of the accept-and-scan method are firstly that it can be achieved without any special extensions to Exim. Furthermore, it's relatively easy to allow detailed per-recipient configuration. For example, some recipients might choose spam scanning whilst others might opt out of it. Specific configurations for each user could be provided; for example, SpamAssassin's rules and scores could be adjusted on an individual basis.

However, accept-and-scan has some *major* disadvantages. By far the biggest one is simply the question of what to do if a message is detected as unwanted (i.e. spam). If your intention is to simply tag or mark the message and perhaps deliver into a separate mailbox, then there's no problem – except that the recipient ultimately still gets the spam/virus. However, if (as is common) you want to reject the message, you are in a sticky situation. You could:

a) Drop (blackhole) it silently – this makes for an unreliable mail system. The sender will never know their mail didn't reach it's destination, and the recipient will have no idea the mail ever existed. OK so long as you are confident that your classification system will never incorrectly decide that a message should be discarded, but otherwise not recommended, and it helps reinforce the impression amongst end-users that e-mail is unreliable (which, in practice, it shouldn't be – so long as it's configured correctly – there are clear hand-offs of responsibility at each stage).

or

b) Create a bounce – this is the "traditional" way for an MTA to deal with mail it cannot deliver, and what commonly ends up happening, but is now very bad practice in the current climate where most senders are faked, except where *absolutely* necessary. The problem is that you will often end up "collateral spamming" innocent third parties who did nothing wrong and were entirely unconnected to the problematic mail. By doing this, you are adding to the problem and effectively shifting the burden of unwanted mail from yourself to someone else. Aside from the fact that the third party victim may well be getting mailbombed with fake bounces if their address is being widely forged, it's likely to be even harder for them to detect the mail as junk and discard it themselves if you've mashed it up by wrapping (and probably truncating) it in a bounce message. However, even if the (faked) sender doesn't exist, you will add load to some innocent party's systems (whoever happens to be running the domain mail exchanger for the faked sender address domain), and your queues will fill up with frozen bounces.

So I must encourage you in the strongest terms to *not* generate bounce messages for unwanted mail. If you decide that accept-and-scan is for you (and I hope to offer compelling reasons why there is a better solution), then please be satisfied with tagging or redelivering messages, rather than bouncing. At worst, ditch messages you don't want – but don't be surprised when your users and their correspondents complain about an unreliable service.

Finally, another disadvantage of the accept-and-scan method is that you may well end up scanning a message multiple times – one for each recipient. This is of course what gives you the flexibility to have per-recipient configurations which is great, but if in fact your users have the same rules then you may expend resources needlessly rescanning mail.

SMTP-time scanning

SMTP-time scanning is a relatively new variation on the creative use of SMTP, but one which has a lot of merits and is gaining support, particularly amongst Exim users due to the ease of integration. The concept is to scan during the SMTP DATA phase, whilst the session is still open and before the receiving server has issued an SMTP confirmation response (at which point it accepts responsibility for the message, leading to the previously-discussed problems with accept-and-scan). If the message is deemed to be unacceptable for whatever reason (looks like spam, contains a virus etc.) then an SMTP code in the 500 series (permanent failure) is issued. There are a number of reasons why this approach has distinct advantages over accept-and-scan:

- Elegance: there is a certain logic in saying that if you're not going to accept a message, it's better to reject it outright at the earliest stage possible rather than accept and then bounce.
- Reduces collateral spam. This is a major consideration, as it minimises the problem of generating bounces as discussed earlier with accept-and-scan. Since you reject unwanted mail at SMTP time, your server never *generates* a bounce message. (Any message that is to be passed to the sender is sent as part of the SMTP response). Now, at *worst* this will mean a similar result to accept-and-scan: the sending server will see the SMTP 5xx rejection, and generate its own bounce, which may well go back to an innocent third party. Superficially therefore, you may fail to see the advantage. ***However***, there are two key points to bear in mind:
  - Above all, a very large amount of spam and viruses are “direct-to-MX”, meaning that the mail is sent directly from the source (typically a spammer with bulk mail software, or a small SMTP engine contained within a virus on a compromised machine). In these cases, the sending client is not a “real” MTA and is almost certain not to generate bounce messages
  - Even if the worst case occurs, and the spam has passed through an intermediate “real” MTA and a bounce is generated, *you* have done all that *you* can to stop being part of the problem. You can't help it if someone else has accepted the “bad” mail and is then forced to generate a bounce, but you have ensured that you are not generating collateral spam yourself. Perhaps eventually the operator of the server the mail passed through prior to reaching you will install the same level of junk protection as you have, thus pushing the problem further and further upstream and closer to the source of the mail. This can only be a good thing.
- No more queues filled with bounces. You shouldn't have undeliverable bounces sitting around on your queues any more except in rare cases where you have been forced to accept a mail which you subsequently pass on via SMTP and it is rejected for some reason. Hopefully you can minimise the instances of this if it affects you using recipient callouts or suchlike, but either way you will have taken a major step towards reducing the problem.

There are, however, some disadvantages that need to be considered.

- Requires enough resources to scan quickly & return back to the SMTP session rather than being able to scan at your leisure according to available resources. The difficulty of this depends on your load and the hardware which you have available. You do not want to unduly delay mail at the DATA stage and this is advised against by the RFCs (see RFC2821 s6.1 and RFC1047). However, many people are finding that scanning and returning a response within a few seconds or so is not a problem – which is insignificant on the scale of things (RFC2821 s.4.5.3 specifies that senders should wait up to 10 minutes for a response, though some servers do not wait nearly so long). If you take an undue amount of time to respond, there is always a slight risk that the message may be delivered more than once due to a synchronisation error – if the sending client times out and keeps retrying a message, whilst in fact the receiving end does deliver the message, but the confirmation does not reach the client in time.
- Per-user configuration options are limited. An important point which is often overlooked by those new to SMTP-time content scanning is that although a message may have multiple recipients within a single SMTP session, there is only *one* opportunity to reply 'yes' or 'no' at the DATA stage. Therefore, content scanning normally only takes place once per *message*, not per *recipient*. (There are some workarounds to this, which we will come to later).

Given the benefits, and the huge problem of collateral spam, we will focus primarily on SMTP-time scanning.

Software Required

We'll now take a look at the software required to implement content-scanning using external software.

The main pieces of software that are required as follows:

- Exim :-)
- Content-scanning “glue” to pass the mail from Exim to the external software and return a result. The Exim local_scan interface was initially intended for applications of this type, and is used by some software such as SA-Exim. However, since v4.50, the standard Exim distribution has included special ACL features (Content Scanning Extensions) which enable the passing of messages to many scanners. Before v4.50, these are available via a source code patch called “Exiscan”.
- Scanning software (virus/spam scanners etc.)
  - Should be daemonised if possible for performance. The overhead of reloading (for example) an entire virus database every time you want to scan a message can be huge compared with a memory-resident daemon which loads the patterns once and handles an indefinite number of 'scan requests' over time.

However, before diving in, consider the *policies* to be implemented as well as the *tools* to do it. There is a lot more subtlety in the infinite variety of policy choices to be made compared to the tools available, though it's also important to decide *what* you want to achieve so that you can choose the right software and methods. For example, a simple policy might state:

- Reject all mails containing attachments that end in .pif/.scr/.bat/.com/.exe
- Reject all mails containing viruses detected by the XYZ virus scanner
- Reject all mails which score more than ten SpamAssassin points
- Reject all of the above at SMTP time

Content-scanning glue: overview

- Exim Content Scanning Extension (from v4.50; formerly called “Exiscan”) – this is the “Swiss army knife” of content scanning for Exim and includes support for a number of external anti-spam/anti-virus tools including SpamAssassin, Sophos/sophie, Kaspersky, ClamAV and Brightmail. It also includes a generic command line scanner plugin allowing arbitrary virus scanners to be used. The Content Scanning Extension is able to operate not only on the content of messages, but decode MIME and uuencoded parts within messages. This can be useful for a number of purposes. Operates in the ACL system.

- SA-Exim – single-purpose spam-scanning patch for SpamAssassin. Extensive & detailed functionality though increasingly most can be done with Content Scanning Extension. Includes 'greylisting', 'tarpitting' and more. Operates using the local_scan system and separate configuration file.

  http://marc.merlins.org/linux/exim/sa.html

Scanning Software

Anti-virus

Clam Antivirus

Clam Antivirus is a package that first started making news a couple of years ago. Originally vaguely inspired by (and sharing some virus definitions with) the Open Antivirus project, it's become the *de-facto* standard Free software virus scanner which has a lot going for it. Whilst as with all Free software projects, it's community-supported and there's no helpline to call, in practice experience with recent virus threats has shown that virus signatures have appeared quickly (even quicker than some commercial virus scanners in some cases! The authors claim that responses are often within an hour of a virus outbreak being recognised) and detection is good.

Clam comes with a daemon (clamd) which is a great help in making it useful without consuming excessive resources. There is also a separate daemon (freshclam) which monitors for virus signature updates and can download and install them automatically.

Some people have reported scalability and stability issues, though many are reported to use it successfully in environments processing a significant amount of mail. A freely available variation on the clamd daemon, called 'nclamd', uses pre-forking processes rather than threads and is claimed to be more robust. The Clam authors did indicate at one point that they would probably merge an option for fork-based processes into the main package at some point, although the tracking down and elimination of some memory leaks seems to have put this idea onto the back burner, and indeed nclamd itself has not been updated since January 2004.

Sophos

Sophos is a commercial virus scanner from a UK-based company of the same name. The usual pros/cons apply: it's closed-source which can be problematic from a practical point of view on open-source operating systems (for example, platforms supported are limited, and you can't recompile if there are library or other dependency issues) as well as (to some people) from a philosophical point of view. It naturally costs real money, but at the same time you get real people you can phone up and speak to.

The software doesn't include a memory-resident daemon, but a widely-used and stable Free software daemon called 'sophie' (http://www.clanfield.info/sophie/) which is dedicated to this purpose works well in conjunction with the shared library provided by Sophos (libsavi).

Others

There are a multitude of scanners out there, both commercial and free, including Kaspersky, ScannerDaemon and others. Take your pick, though it's worth considering which are supported directly by Exim's Content Scanning Extension if you're planning on using that.

Anti-spam

SpamAssassin

By far the most common tool in use today is the ubiquitous SpamAssassin, which you will almost inevitably have heard of. SpamAssassin is a Perl-based scanner which includes both command line tools and a daemon for checking e-mail for signs of spam.

It primarily consists of a large base of pattern-matching rules, but includes a lot of other checks such as checking mail servers that the mail has passed through against DNS blacklists or even comparing the content against distributed checksum databases such as Razor. Most powerful, however, is SpamAssassin's Bayesian learning techniques, which allow you to “train” it what is spam and what's not, and it will use the data you give it in predicting the likelihood of future mails being spam or not. The Bayesian system alone is extremely effective if trained well, even in the face of “Bayes poison” attacks where spammers include lots of garbage words in their e-mails (often hidden) in order to try to disrupt the statistics behind Bayes techniques.

The basic premise of SpamAssassin is a scoring system, where each “rule” which is hit by a particular mail adds a certain “score” to the total. The scores added for each rule vary considerably and can be customised. Decisions can then be made on what to do with the mail according to the score. Typically, the scores are used to “flag” mails as being possible spam if they exceed a modest score (say, 5) and reject mails outright that exceed a high score (say, 10-12). “Flagging” can consist of changing the Subject line, adding extra headers to the mail or even rewriting the body to warn the recipient that it was detected as spam and include the spam as an attachment (the latter works with SA-Exim but not the Content Scanning Extensions). Note that SpamAssassin itself does not implement any of the policy (e.g. rejection); it merely analyses the mail.

One thing worth bearing in mind is that SpamAssassin's popularity is also its greatest weakness: whilst it was extraordinarily effective back in time, it's widespread use and the growing sophistication of spammers means that many spammers are now good at crafting their junk such that it evades SpamAssassin's filters. This not only makes the Bayesian learning all the more important, but also means that you should be prepared to at the least keep an eye on the “add on” rulesets that are available (many superb ones at http://www.rulesemporium.com) and probably be prepared to write your own rules from time to time (this is very easy if you are familiar with regular expressions).

Others

Partly due to the problems of spammers evading SpamAssassin, interest in alternative solutions is growing. Other tools that can be used include spamprobe, bogofilter, dspam and CRM114. Whilst it's certainly possible to do useful things with these in conjunction with Exim, none of them are quite so easy to integrate as SpamAssassin at the current time.

Exim Content Scanning Extension

We'll look at Exim's built-in Content Scanning Extension in a little more detail now, since it's so important if you're considering content scanning with Exim.

As mentioned earlier, it was originally a separate project called Exiscan, but has been merged since v4.50. It is still maintained by its original author, Tom Kistner of Astaro Internet Security. To enable it, you must compile Exim using the build-time WITH_CONTENT_SCAN option enabled in your Makefile. You may also want to use the WITH_OLD_DEMIME option, discussed later. These options (or at least WITH_CONTENT_SCAN) are frequently enabled in binary distributions (RPMs, DEBs etc.)

Unlike some other solutions such as SA-Exim, the Content Scanning Extension bypasses the local_scan interface provided by Philip to hook into the message processing system, and instead integrates tightly with Exim by hooking into the ACL system.

So, what does the extension actually provide?

- New options in DATA ACL to call external scanning software.
- Inbuilt MIME decoder. Whereas Exim as an MTA itself deals with message bodies solely as whole parts with little regard to the semantics of them, the Content Scanning Extension can decode individual parts, attachments etc. and take action based on that.
- MIME checking to detect serious MIME errors (often indicative of malware) – MIME decoders in mail user agents vary considerably in their tolerance of malformed MIME. This has been exploited in the past by malware, so blocking some
- File extension matching (e.g. to block all .pif files). There is a specific option in the Content Scanning Extension called “demime” to serve this purpose, although with recent versions it has been generalised with the provision of an attachment filename variable and the MIME ACL. The “demime” variant is officially deprecated and needs to be specifically enabled using the WITH_OLD_DEMIME build-time option, but is planned to remain in Exim indefinitely.
- Regular expression matching of decoded or raw MIME parts.
- New ACL: acl_smtp_mime – called once per MIME part. Unusual in the ACL system, it doesn't correspond directly to a particular stage of the SMTP transaction; rather, there is an

interaction with the DATA command. The MIME ACL is run before the DATA ACL, and a 'deny' in the MIME ACL will cause the DATA command to be rejected outright.

The MIME ACL provides a number of new variables including $mime_content_type, $mime_filename and many more. These can be used for any purpose you can conceive; the most obvious is file extension blocking as mentioned but also, for example, some viruses have been included as attachments with fixed (and unusual) filenames, so you could use a condition involving the $mime_filename variable to reject them.

Some brief Content Scanning Extension examples

As with the Exim ACL system in general, there is an enormous amount of flexibility and therefore the precise details of how you use it will vary considerably according to exactly what you want to do. The best thing to do is use the documentation & examples from the Exim manual and decide what your precise policies are. Nevertheless, to give a brief taster of some of the simple uses:

Reject spam

A simple example of calling SpamAssassin as user 'nobody' from the DATA ACL and limiting the scanning to messages under 80kB would be as follows:

```
deny message   = This message was classed as spam
     condition = ${if <{$message_size}{80k}{1}{0}}
     spam      = nobody
```

This will, however, reject all messages marked as spam by SpamAssassin. In practice you'll probably want to make use of the $spam_score_int variable (which contains the SpamAssassin score multiplied by 10) to only reject mail over a certain score, allowing “grey area” mails which look like spam but only scored a modest number of points to pass. For example, a condition which would reject mails with a spam score of over 9.9:

```
condition = ${if >{$spam_score_int}{99}{1}{0}}
```

Reject viruses

The use of this will vary according to whether you have a virus scanner that has a reliable internal MIME decoder. Many do, in which case you can just pass the entire message body to it using the DATA ACL. (If you are using ClamAV, enable the ScanMail option). If not, you can use the Content Scanning Extension's in-built MIME decoding in the MIME ACL.

```
deny message   = Message contains a virus ($malware_name)
     malware   = *
```

MIME checking

The `$demime_reason` and `$demime_errorlevel` variables can be used in the detection and rejection of malformed MIME messages. An example, used in the DATA ACL would be:

```
deny message   = Serious MIME defect detected ($demime_reason)
     demime    = *
     condition = ${if >{$demime_errorlevel}{2}{1}{0}}
```

Other considerations

There are a couple of problems in particular that are often raised in conjunction with content scanning. We'll take a brief look at a couple of them, along with an example of a common usage scenario.

The MX problem

The “MX problem” is a problem introduced where you have more than one public-facing mail server, and the mail servers do not give identical responses to a similar SMTP session. For example, if one mail server has DNSBL lookups and content scanning, and the other does not.

The main potential problems are:

- Creating a spam 'back-door'
- Creating collateral spam

This problem is not specific to content scanning, since it applies to any other policy rejection that takes place at SMTP time (including, for example, DNSBL lookups applied at RCPT time and even rejection of invalid recipients), it's worth a brief mention since the implementation of content scanning often goes alongside more stringent SMTP-time checks in general, and these usually overshadow the “invalid recipient” case by some amount.

Creating a spam back-door

Consider the case of an e-mail which is sent to your primary MX server. Let's assume it was rejected at RCPT time due to the sender being listed on a DNS blacklist. Or, perhaps, even if it passed that check, it might be rejected at DATA time due to SpamAssassin rating, perhaps taking into account a number of factors including servers which the mail passed through before reaching yours (based on the Received headers), and the content of the mail.

Now, consider the same mail sent to a “dumb forwarder”; a traditional “backup MX” in other words, which accepts any mail sent to the domains it is configured to handle. The server would accept the mail, and then forward it to the primary MX. However, the primary MX has now lost the ability to do meaningful DNSBL checks, because the “source” is now no longer the spammer's machine but your (trusted) secondary mail server. Similarly, it's conceivable that the SpamAssassin rating might be different for the same reasons, perhaps pulling the mail under the threshold for rejection.

So, what this means is that you have created a “back-door” to your mail system: a mail sent to your “backup MX” rather than your primary server may now reach it's intended recipient even though it would normally have failed. Some spammers are known to exploit this common mistake by missing the primary MX altogether and targeting secondary MXes.

Creating collateral spam

The problem doesn't end there. Even if your primary MX manages to reject the unwanted mail, perhaps at DATA time, your secondary MX is left in the situation that we discussed earlier – it has “spam in hand” and has to create a non-delivery report for it. If the sender of the spam is forged, this means creating collateral spam to an innocent third party – again, bad practice.

The solution

Firstly, consider whether you really need multiple MXes. If you only have one “ultimate endpoint” machine, then multiple MXes don't generally give a lot of benefit unless your “ultimate endpoint” is inaccessible to the Internet at large for long periods of time (say, several days). Although having a “backup MX” does mean you can accept mail whilst the primary server is down, and perhaps then receive that mail more quickly when the endpoint server comes back up (since you can flush the queue on the secondary, or have an aggressive retry pattern), this is often a relatively small benefit compared to problems you cause.

If you must have more than one MX, then the key point to remember is that you should engineer your systems such that *every mail server gives out identical responses*. Protect each machine with equal strength to ensure that you don't introduce back-doors or generate collateral spam.

The multiple-recipient problem

Hopefully you will have been inspired by the methods discussed today. However, one practical question which is often raised as soon as people try to implement content scanning is this: “How do I allow users to have different preferences or opt in/out of scanning?”.

The answer, unfortunately, is not simple. For SMTP-time scanning, we are limited by the constraints of the SMTP protocol. As mentioned earlier, although a message may have multiple recipients, you can only return one result to the DATA command which will apply to all the recipients. Clearly, as a limitation of the protocol, there is no easy way around this problem. However, there are reasonable workarounds and it therefore should rarely be an obstacle in the way of implementing content scanning at SMTP time.

The first thing to do is make sure you're solving the “right” problem. Why is it that some users need different scanning options? Do they *really* need different scanning options, or are there (global) changes that you could make so that everyone is happy? If not, and if the options are on a per-domain basis rather than per-user, could you set up a separate server for customers/users/departments that don't want their mail scanned, or want a different (global) option? (You would then point the MX records for the domain in question to the other machine).

Failing all that, if you decide you really do need to implement multiple scanning options on a single machine, the main thing to do is to try to limit the number of variations, as this will make workarounds more viable. Even if “one size doesn't fit all”, maybe “two sizes” do? (e.g. Scanning on/off).

The main workaround is to defer one or more recipients in each SMTP session, so that each session only contains recipients with the same scanning preferences. There are a number of ways to do this. The most simplistic is obviously to restrict each SMTP session to a single recipient, and defer subsequent recipients using the RCPT ACL. This is probably not a good idea; consider the case of a message with 10 recipients; the sending server will deliver the message to one recipient, wait for its retry time, deliver to the second and so on. It could be a considerable time before all 10 recipients have received the mail. Add more recipients and the sender might even give up before deliveries to all recipients are complete!

A more intelligent approach is the concept of “scan profiles”. With this approach, the first recipient in a session defines the scan preferences for that session. Subsequent recipients in that session which have the same preferences are accepted; others are deferred. Clearly, this is where the number of variations on scanning offered has an important effect: if only two “scan profiles” are offered, *at most* a mail will be delayed to *some* of its recipients by a time period equal to the retry time of the sender. Usually, this will be acceptable especially when you consider that, depending on your organisation, the percentage of messages with multiple recipients *and* where the recipients fall into more than one “scanning profile” is likely to be small. However, if you have lots of “scan profiles”, delays could potentially be more significant, and unacceptable.

- See http://www.exim.org/pipermail/exim-users/Week-of-Mon-20031006/061151.html

A common usage scenario: Exim as a transparent front end

Another question which is raised very frequently is how to use Exim as a “transparent” front end to an existing mail system, using Exim to do all policy rejections including content scanning and pass the messages on to an existing internal system. (The same applies for those wanting to provide “outsourced” content scanning).

The answer is very simple. All you need to do, typically, from a default configuration, is add the destination domain(s) to the domainlist `relay_to_domains`, and then, for the specific domains that are to be handled manually, have a router using the 'manualroute' driver which routes them manually to the “real” destination server. This router will, naturally, need to go before the general “dnslookup” router. A simple, 'static' example might be as follows, if the domain “somecompany.example.com” was to be routed to an internal mail server with IP address 192.168.0.1:

```
route_scanned_mail:
   driver = manualroute
   domains = somecompany.example.com
   route_data = 192.168.0.1
   transport = remote_smtp
   no_more
```

A similar but more general method might contain a list of domains and their 'manualroute' destinations in a DBM file:

```
route_scanned_mail:
   driver = manualroute
   domains = dbm;/etc/exim/domain_routes
   route_data = ${lookup{$domain}dbm{/etc/exim/domain_routes}}
   transport = remote_smtp
   no_more
```

Conclusions

- Content scanning is a useful tool as part of a wider policy framework. Some kind of content scanning will almost certainly bring benefits, but don't use it to the exclusion of other methods.
- Content scanning needs responsible planning and implementation to avoid amplifying the problem or moving the burden to someone else. Scanning and rejection at SMTP time is highly recommended and even if you choose not to use this method, *please* don't generate collateral spam by bouncing mail you've accepted and then decided you don't want.
- The Content Scanning Extension to Exim is integrated into the Exim distribution, widely used, stable and powerful, and allows scanning at SMTP time for:
  - Anti-virus purposes, using a variety of scanners both free and commercial. Remember to use a daemon if possible!
  - Anti-spam purposes, using SpamAssassin.
  - File extension blocking – a “cheap”, relatively unintrusive and surprisingly effective “first-line” of defence.
  - Regular expression blocking – but use with care! Blocking an entire mail based on one or two phrases is risky – a system like SpamAssassin's scoring is more forgiving.
  - Almost anything else you can think of based on the content or MIME information contained within a message

To summarise, with so many easy, powerful and free solutions, at the very least *some* basic content scanning (e.g. file extension blocking) is highly recommended and can be achieved with modest resources.

These notes, any corrections or addenda, and Spam & Virus Scanning mini-HOWTO available at:

http://www.timj.co.uk/computing/software/exim/